ARL-TR-7827 • Sep 2016

US Army Research Laboratory

Efficient Multi-Concept Visual Classifier Adaptation in Changing Environments

by Maggie Wigness, John G Rogers, Luis Navarro-Serment, and Bruce A Draper

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-7827 ● SEP 2016

US Army Research Laboratory

Efficient Multi-Concept Visual Classifier Adaptation in Changing Environments

by Maggie Wigness and John G Rogers
Computational and Information Sciences Directorate, ARL

Luis Navarro-Serment
The Robotics Institute at Carnegie Mellon University, Pittsburgh, PA

Bruce A Draper
Colorado State University, Fort Collins, CO

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2016	Technical Report	April 2015-March 2016

4. TITLE AND SUBTITLE
Efficient Multi-Concept Visual Classifier Adaptation in Changing Environments

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS
Maggie Wigness, John G Rogers, Luis Navarro-Serment, and Bruce A Draper

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CII-A
2800 Powder Mill Rd
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-7827

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Multi-concept visual classification is emerging as a common environment perception technique, with applications in autonomous mobile robot navigation. Supervised visual classifiers are typically trained with large sets of images, hand annotated by humans with region boundary outlines followed by label assignment. This annotation is time consuming, and unfortunately, a change in the environment requires new or additional labeling to adapt visual perception. The time it takes for a human to label new data is called *adaptation latency*. High adaptation latency is not simply undesirable, but may be infeasible for scenarios with limited labeling time and resources. We introduce a labeling framework that significantly reduces adaptation latency using unsupervised segmentation and clustering in exchange for a small amount of label noise. We demonstrate the framework's speed and ability to collect environment labels that train high-performing, multi-concept classifiers in several outdoor urban environments. Finally, we show the relevance of this label collection process for visual perception as it applies to navigation in outdoor environments.

15. SUBJECT TERMS
semi-supervised labeling,unsupervised hierarchical clustering,visual classification,autonomous navigation and path planning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maggie Wigness
Unclassified	Unclassified	Unclassified	UU	50	19b. TELEPHONE NUMBER (Include area code) 301-394-3927

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Accurate environment perception is critical for autonomous robots to plan paths on traversable terrain and avoid object collision during navigation. While many sensors have been used to help with perception,[1–4] speedups in image processing have allowed vision-based perception to emerge in mobile robots.[4–10] Visual classification is an important task for many applications, but is particularly useful for path planning because visual data allow robots to perceive a large area of the environment at once. However, a variety of challenging properties associated with visual data can make it difficult to train classifiers. These challenges may include changes in illumination, scale, perspective, color, and background clutter.

Classifiers can learn to account for these factors, but generally need large amounts of training data to sample and model these variations. While collecting visual data is a trivial task, the raw data contain no label information from which supervised classifiers can learn. Label collection is a burdensome task for human annotators as it requires human intervention to assign semantic labels to training instances, and unfortunately, may not be a one-time event. For example, to ensure the highest quality visual perception for mobile robots, training data should be collected from the environment where navigation tasks will be performed. Thus, each domain change requires new data collection and labeling.

Furthermore, state-of-the-art deep learners[11,12] now rely on millions of images for learning. Label collection for the ImageNet data set[13] was performed in parallel via crowdsourcing, but took a combined total of approximately 19 person-years.[14] This process is even more demanding for scene labeling classifiers,[15–17] because distinct regions in images must be outlined before assigning labels. This is infeasible for applications with limited labeling time and resources. We define the time for a human to label a new set of training data as *adaptation latency*. In our robot navigation example, this represents the time robots are unable to navigate autonomously because perception models are being adapted.

To keep up with the demand for labeled training data, more semi-supervised and unsupervised label collection techniques need to be developed to help reduce the overall labeling workload. Specifically, we have identified 4 objectives important in the design of such label collection techniques.

1) **Learning a label set**: Since the training data are initially unlabeled, we assume the set of visual concepts in the data is also initially unknown. The process of learning a label set, commonly referred to as ***concept discovery***,[18–20] determines the classes a classifier will learn to recognize. Only concepts with labeled training examples can be accurately accounted for at test time.

2) **Reducing the workload**: The motivation of this work, reducing labeling effort, is an objective itself. We refer to the degree in which labeling effort is reduced as ***labeling efficiency***. Labeling efficiency is discussed with respect to 1) the overall effort required to label training data (i.e., user interactions) and 2) the overall time required to label training data.

3) **Maximizing label counts**: Trivially, reducing labeling effort can be achieved by simply labeling fewer training samples. However, a small labeled set may not be sufficient to train high-performing classifiers. We refer to this objective as ***exploitation*** of the training data.

4) **Maintaining accuracy**: There is always the possibility for error when humans assign labels to visual data. For example, *gravel* may be mistaken for *asphalt* or a user may mistype *car* as *cat*. The fraction of nonerroneous labels is defined as ***label accuracy***, and the fraction of label errors represents ***label noise***. High label accuracy is important because noise creates confusion during classifier training. Beyond human error, most label noise discussed in this report is introduced by frameworks that employ group-based labeling to improve labeling efficiency. Visual data believed to represent the same visual concept are grouped together, and the entire group is assigned a single label to describe the data (see Fig. 6). When the group of images in fact represents multiple classes, label noise is introduced.

Each objective plays a critical role in determining the success of classifier training. However, the frameworks that have emerged to help alleviate labeling effort tend to focus on a subset of the labeling objectives we laid out instead of working toward all 4. Active learning frameworks[21–25] reduce the workload by labeling only a subset of samples, so of course they do not maximize the label count. Moreover, active learning systems typically assume the label set is known in advance, and run the risk of increasing the total work time by introducing latency while classifiers are retrained. Group-based labeling techniques such as partitional clustering,[26–28]

incremental clustering,[18,29] active clustering,[30,31] and topic modeling[32,33] reduce the workload by labeling groups instead of instances, but can suffer from high label noise or reduced efficiency if the label noise is removed.

Unfortunately, adaptation latency has yet to be discussed in existing supervised multi-concept visual perception systems used in robotics applications.[1,5–7] Annotation of images is performed as a necessary but time-consuming step to train supervised classifiers. Unsupervised or self-supervised approaches have been used to eliminate labeling effort,[3,10,34–37] but produce a limited environment vocabulary (e.g., *traversable* vs. *non-traversable*). These techniques do not generalize well to more complex navigation tasks that require a richer set of scene semantics, such as verbal navigation commands from humans.[38]

Our work is motivated by scenarios that need more than a binary understanding of environments, and that have limited time and resources to collect this information. We discuss our labeling framework designed to model and balance each of the 4 objectives to yield fast label collection that trains high-performing visual classifiers. Specifically, our approach is a group-based labeling technique where groups are selected from a hierarchical clustering of the data. By maintaining a hierarchical clustering, our approach establishes a space of groupings that map to coarse and fine-grained visual concepts. This allows the system to search the hierarchy and discover groups that match the concept granularity of the classifier and thereby keep label noise to a minimum. These groups are identified by searching for local structural changes in the hierarchy. This selection heuristic is combined with criteria that reward exploration of the search space to discover new visual concepts and labeling large clusters to maximize efficiency and total label count. Overall, these measures model our defined objectives, identify clusters from the hierarchy that can be labeled with little effort, produce minimal label noise, and most importantly collect data that train high-performing visual classifiers.

Using several outdoor urban environments, we show that visual perception trained with our efficient label collection technique allows for reliable path planning and successful navigation. We compare the approach to a fully supervised labeling approach by evaluating pixel labeling rate, pixel-wise classification, and autonomous navigation via road terrain with respect to adaptation latency.

2. Related Work

2.1 Label Collection

There are 3 dominant approaches used to address the labeling workload problem: crowdsourcing, active learning, and group-based labeling. Crowdsourcing via marketplaces such as Amazon Mechanical Turk has become a popular way to collect large sets of annotated visual data.[13,39] This technique allows the label collection process to be split into smaller units of labor, and these tasks are distributed to a set of human resources who work in parallel. Crowdsourcing has several shortcomings. First, this approach can be quite costly for data sets with millions of images since users are paid per labeling task. Second, it has been found that users are highly inconsistent in their labeling.[40] These labeling inconsistencies require reconciliation and verification steps that also require human effort.

Active learning is an instance-based labeling approach that tries to identify an informative subset of training samples to label with a supervised learner in the loop. Selection criteria include uncertainty sampling,[22–24] Gaussian process models,[21] and information density.[25] Active learning reduces the number of image instances a user must label, but often requires a priori knowledge for classifier seeding and introduces latency while iteratively retraining classifiers. Active learning has been combined with crowdsourcing,[41,42] but Vijayanarasimhan et al.[41] note that retraining classifiers after each labeling queries creates latency that makes the parallelism of crowdsourcing less evident.

Group-based labeling reduces workload by providing a single label to a group of samples. Clustering[26–28] and topic modeling[32,33] form groups through bottom-up discovery, requiring no a priori knowledge of the unlabeled data. These techniques try to find a 1-to-1 mapping between groups and visual concepts. Unfortunately, visual data properties make partitioning the data difficult and groups often contain data from multiple classes. Assigning the dominant class label to an entire set of images that represent multiple concepts can create label noise.

Label noise can be reduced at the cost of additional labeling effort and labeling latency. Active clustering improves group coherency by iteratively collecting constraints indicating whether 2 images represent the same class.[30,31,43] This constraint information is used to augment feature representations and recluster data. Lee and

Grauman cluster the “easiest" subset of unlabeled data and label a single group on each iteration to improve overall group coherency.[18] A largest subset labeling approach is used by Galleguillos et al. in their iterative labeling approach with multiple kernel metric learning.[29] Largest subset labeling eliminates label noise by asking a user to remove images from a group that do not represent the dominating class label. Each of these techniques introduces reclustering latency after each labeling query.

Related, there has been work on how to reduce labeling effort for video data. Xie et al. introduce a label transfer approach where coarse 3-D annotations of street scenes can be transferred to 2-D images.[44] Other semi-supervised label propagation for video streams has also been achieved with random forests[45] and a mixture of temporal trees.[46] These approaches use the information encoded by temporal consistency to reduce labeling effort, but are not compatible for large sets of nonsequential training images (e.g., environment A in our experiments).

2.2 Visual Perception in Robotics

Vision provides valuable perception for mobile robots. Terrain and obstacle classification are particularly important to help determine traversability. For example, visual terrain classification has been used to identify when legged robots should change gaits,[6,7] and aerial robots can identify possible landing sites or be used to communicate with ground robots when working in teams.[5] Visual perception is also being used for path planning on ground robots. Haselich et al. fuse 3-D laser scans and camera images to perceive *road*, *rough*, and *obstacle* terrain classes.[1] Haselich et al. is the first to mention the inability to adapt quickly to new environments due to the requirement of reannotation.

Consequently, a significant amount of visual perception path planning research focuses on semi-supervised, self-supervised, and online learning. Teleoperation has been used to define optimal routes to infer *path* and *nonpath* labels for visual classifiers.[47] Ross et al. identify obstacles with an unsupervised, online technique that compares visual appearance and structure to learned environment models.[10] Roncancio et al. adapt a pretrained supervised visual classifier online to identify *traversable* and *non-traversable* paths.[9]

Other techniques pair vision with complimentary sensors. Visual features have been used to enhance radar *ground* prediction.[3] The correspondence between visual fea-

tures and a robot's navigation experience (e.g., slippage) was used to identify *traversable* terrain.[34] Lookingbill et al. used a reverse optical flow technique to update visual classifiers with the appearance of obstacles beyond the range of stereo vision.[36] Other self-supervised learning examples include combining vision and LiDAR.[35,37]

These examples adapt terrain classifiers without the time-consuming labeling process. However, the lack of human supervision has limited most of this work to binary classification (e.g., *traversability*). Unfortunately, these approaches do not extend to more complex multiclass tasks such as verbal navigation commands from human to robot.[38]

3. Reducing Adaptation Latency

Our labeling system is designed to be quick and efficient so new sets of labeled training data can be easily collected by a single human annotator. Our approach, called hierarchical cluster guided labeling (HCGL), iteratively selects clusters to label from a hierarchical clustering of the data samples. Before motivating our use of hierarchical clustering and discussing details of our group selection criteria, we overview the traditional supervised labeling approach used to label environment images.

3.1 Supervised Labeling

Supervised label collection produces high quality labeled data, but is time consuming for 2 reasons: 1) training sets are typically large and 2) images capture multiple terrains and objects in the scene that need to be localized before label assignment. Image annotation tools such as LabelMe[48] have been used to facilitate supervised labeling. LabelMe allows annotators to precisely outline, via mouse clicks, and assign labels to each distinct region. Figure 1 is an example of a training image (left), required outlining (middle) and labeled output (right; see class/color legend in Fig. 18) using LabelMe. Labeling 250 images requires over 20 h of effort (discussed in Section 4), causing high latency during domain changes and inhibits fast adaptation.

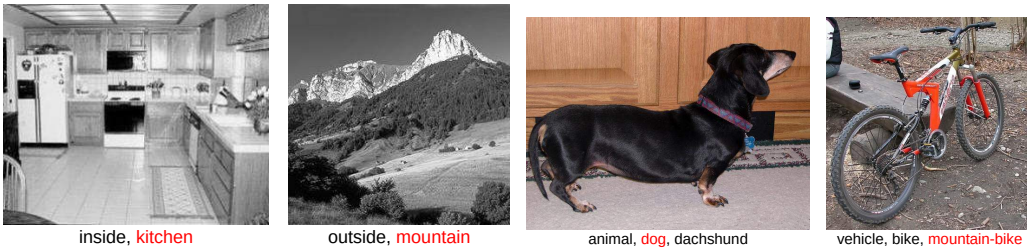
Fig. 1 Example image, outline annotation and label results of a supervised labeling process

The goal of this work is to train supervised multi-concept visual classifiers using large amounts of labeled environment data with limited human interaction. We use the HCGL framework,[49] originally designed to cluster and label groups of single-concept images. We discuss HCGL and several modifications made to the framework to better suit real-world environment data. After discussing the efficient label collection technique, we compare HCGL to supervised labeling with LabelMe to demonstrate the speedup achieved.

3.2 Hierarchical Motivation

As previously mentioned, many group-based labeling techniques have emerged. One major disadvantage of group-based labeling is the addition of label noise when images in the same group represent multiple visual concepts. We hypothesize that label noise collected by partitional group-based labeling approaches is caused in large part because the unsupervised grouping algorithm learns feature patterns that map to a different concept granularity than the concepts of interest for the classification task.

Visual concepts are hierarchical. Figure 2 includes example images from 2 single-concept benchmark classification data sets, 13-Scenes[50] and Caltech-256.[51] The labels in red indicate the concept granularity these benchmark data sets would use to evaluate supervised classifiers. However, note that all labels associated with an image are valid visual concept descriptions, and represent a progression of descriptions from coarse-grained to fine-grained.



(a) Scene images and labels **(b) Object images and labels**

Fig. 2 Examples of coarse- and fine-grained visual concepts for benchmark data images, where labels in red indicate the label used for benchmark classification evaluation

The label set of interest to a classifier (which is task dependent) is denoted as $\mathcal{Y}$, where $|\mathcal{Y}| = K$. In the object labeling example, *dog* $\in \mathcal{Y}$. The goal of partitional grouping techniques is to find a partition of K groups such that each group maps to a label in $\mathcal{Y}$. Grouping is influenced by feature representation and intraclass and interclass similarity, which are hard to manage explicitly with an unsupervised grouping algorithm. In many cases, a grouping algorithm may identify a pattern in the data that represents a coarser- or finer-grained concept than those defined in $\mathcal{Y}$. For example, in the 13-Scenes data set groups of images representing *coast* and *highway* share a coarse-grained *open* quality since the horizon is visible in both classes. Alternatively, images might be grouped at the fine-grained level of *dog*, *cow*, and *sheep* when the task is only interested in *animal*.

Instead of forcing groupings to occur at a particular level of granularity, we use hierarchical clustering to maintain a spectrum of pattern similarities encoded in image groupings. Figure 3 illustrates this approach with 5 classes from the 13-Scenes data set. We denote the hierarchy as $\mathcal{H}$. Nodes colored black correspond to groups that contain images from multiple scene classes. The remaining colors indicate groups of images from a single scene class. There is an obvious division of the hierarchy into 4 groups: *tall building* (green), *living room* (blue), *suburb* (yellow), and the coarse-grained concept of *open* (dashed outline) previously mentioned. The many smaller, interweaved partitions of the *coast* (red) and *highway* (orange) classes, as subtrees of the *open* partition, are evidence of high interclass similarity.

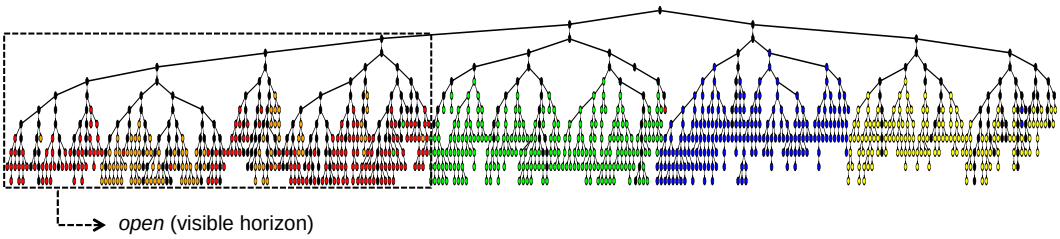


Fig. 3 Hierarchical clustering of 5 classes from the 13-Scenes data set. Node colors indicate what class the images in the cluster represent: yellow-*suburb*, blue-*living room*, green-*tall building*, red-*coast*, and orange-*highway*. The section of the hierarchy outlined by the dotted line highlights the data grouped as the coarse-grained visual concept *open*.

Maintaining the hierarchical structure provides many unique benefits to our labeling framework. First, the number of clusters does not have to be known in advance. Instead, the hierarchy allows each class to breakdown coherently at its own pace (i.e., at different levels in $\mathcal{H}$). Second, the hierarchical relationships provide information about how feature patterns change as data are further refined into smaller groups. Later in this section, we describe how we use these relationships to define *interestingness* and guide our search for clusters that should be labeled. Third, the hierarchical clustering avoids latency during labeling. If the system asks the user to label a cluster that is too high in the tree (i.e., too coarse-grained for the label set $\mathcal{Y}$), the user marks it as *too coarse* and the system immediately has access to the subtree below it, representing possible finer-grained concepts. No reclustering is necessary, and no classifier has to be retrained while the user waits.

3.3 Iterative Selection and Labeling

Hierarchical clustering is not the solution to the label collection problem but rather an encoding of information that the HCGL framework uses to solve the labeling problem. At a very high level, HCGL is simply an iterative group-based labeling technique. That is, HCGL selects a group from the hierarchy and displays the images in the group to a user who assigns it a label. This process repeats until there are no more groups to be labeled or, more likely, the user runs out of time.

Figure 4 illustrates a single labeling interaction in the iterative HCGL process and its use of majority group-based labeling. Since most images in the group are of a *dog*, the group is labeled as such. The images in this group that do not represent *dog* become examples of label noise. Annotators are encouraged to only supply labels

to groups of images that are dominated by a single label in $\mathcal{Y}$, to keep the level of noise down. When HGCL mistakenly selects a group that is not dominated by a single label, the annotator labels it as *too coarse*, telling HCGL that it selected a node too high in the hierarchy for this branch of the tree.

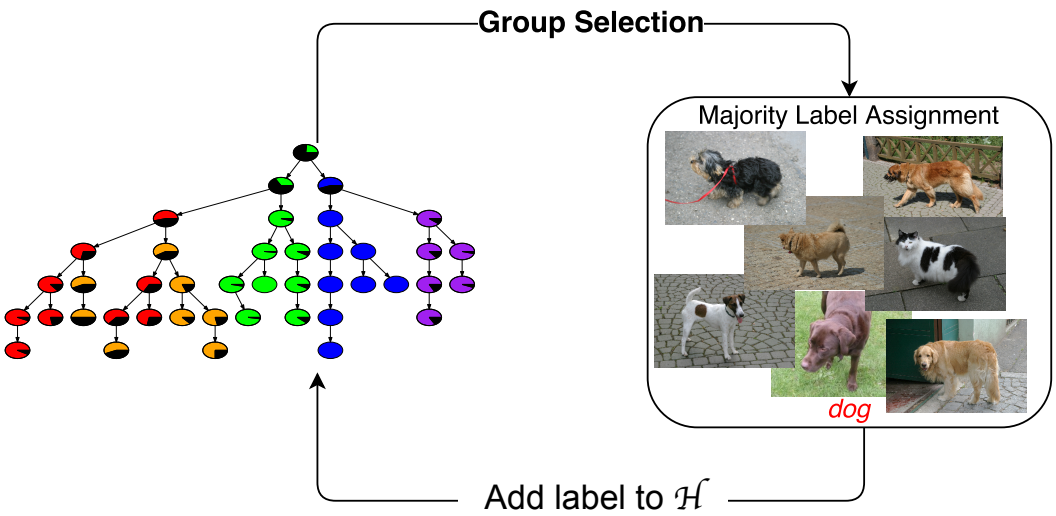


Fig. 4 Illustration of a labeling iteration performed by HCGL

We note that the iterative labeling process may terminate before all the training data have been labeled. This framework is designed around the assumption that in many real-world applications, labeling a large set of training data in its entirety is not feasible. Thus, the group selection technique employed by HCGL is a primary contribution of this work. Groups need to be selected in such a way that a diverse and accurate set of training data can be collected even when users only have a small amount of time to devote to labeling.

Group-based labeling is beneficial for these types of real-world applications since multiple images are labeled with a single labeling query, leading to efficiency gains. However, unlike partitional grouping approaches, which create a set of disjoint groups and ask the user to assign a label to each group, HCGL must select groups from $\mathcal{H}$, which contains nondisjoint groups and some redundant information. For example, if cluster c in Fig. 5 contains *dog* images, it must be true that its children, c_r and c_l, represent the same concept since they each contain a subset of images represented in c. Thus, the selection order of groups in $\mathcal{H}$ is meaningful because descendants of a labeled group (according to the structure in $\mathcal{H}$) can inherit labels

without being viewed and labeled by the annotator. The remainder of this section focuses on the novel technique HCGL uses to select groups from $\mathcal{H}$ during the iterative labeling process.

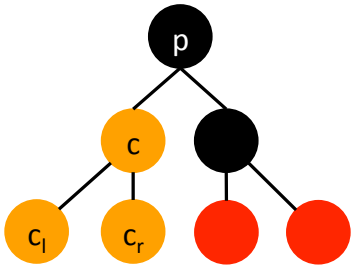


Fig. 5 Illustration that depicts the relationships for group c in a local neighborhood of $\mathcal{H}$, including its parent p and left and right children, c_l and c_r

3.4 Group Selection

HCGL group selection is designed to balance the labeling objectives of discovery, efficiency, exploitation, and accuracy. To do this, we define the following heuristic criteria for groups in $\mathcal{H}$:

1) Interestingness: the degree of structural change seen after a split in $\mathcal{H}$
2) Exploitation: the number of samples that would receive labels
3) Exploration: the likelihood that a group represents a different concept from those previously labeled

These 3 criteria are discussed individually in detail, followed by a discussion of how the criteria are combined to create our novel group selection criteria.

3.4.1 Interestingness

The hierarchy encodes groups of data that span a spectrum of concept granularities, but the classification task, and thus the labeling task, focuses on a specific but initially unknown set of labels $\mathcal{Y}$. This means that for HCGL to be successful, the algorithm must find locations (image groups) in $\mathcal{H}$ that are most likely to represent a single label in $\mathcal{Y}$. As stated previously, we assume no a priori knowledge of $\mathcal{Y}$. Instead, HCGL compares the image features in hierarchically related groups to measure local structural change. More specifically, ***interestingness*** is defined as

the degree of change at a split in $\mathcal{H}$. The idea is that feature similarities encoded in $\mathcal{H}$ map to coarse- and fine-grained visual concepts. When underlying patterns of similarity change, it is likely that a visual concept transition has also occurred.

HCGL compares the structural change between a cluster, c, and its parent, p (this relationship can be seen in Fig. 5). The internal structure of c is derived from its data matrix, X_c, where each column is an image represented by a d-dimensional feature vector:

$$X_c = \begin{pmatrix} x_{1,1} & x_{2,1} & \cdots & x_{s,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{s,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,d} & x_{2,d} & \cdots & x_{s,d} \end{pmatrix} \tag{1}$$

The data are mean centered, $\hat{X}_c = X_c - \bar{X}_c$, and the covariance matrix of $\hat{X}_c$ is decomposed and represented by its eigenvectors, V_c, using singular value decomposition:

$$V_c \, \Lambda_c \, V_c^{-1} = SVD(Cov(\hat{X}_c)) \tag{2}$$

$$= SVD\left(\frac{\hat{X}_c \hat{X}_c^{\,T}}{s-1}\right) \tag{3}$$

This representation of internal structure models the direction of variance in the data. Given that the diagonal entries of Λ_c are sorted in descending order, the first eigenvector, v_{c1}, in V_c provides the axis of maximum variance for c.

The structural change between c and p is calculated as the angle between the first eigenvectors, v_{c1} and v_{p1}, of c and p, respectively. Larger angles indicate greater differences in directions of variance and are therefore more interesting. Formally, interestingness derived from structural change for group c is defined as the cosine distance,

$$\Delta(c) = 1.0 - \langle v_{c1}, v_{p1} \rangle, \tag{4}$$

which yields values on the interval of $[0.0, 1.0]$ with large Δ values representing large angles. The idea behind this type of selection is to order groups by the strength of their potential concept transition.

3.4.2 Exploitation

Exploitation selection is based on the number of unlabeled samples in a cluster. This criterion is designed to label larger clusters first to emphasize the efficiency objective and collect a large set of labeled data quickly. In other words, cluster c in Fig. 5 has a higher exploitation value than both c_l and c_r since each contain a subset of c's images, resulting in fewer labeled images per labeling query.

However, the exploitation value is not necessarily equal to the number of images in a group, because some of the descendant groups may already have been labeled in a previous iteration of HCGL. This set of previously labeled groups is denoted as $\mathcal{L}$. Formally, the exploitation score for group c_i is defined as

$$\xi(c_i) = |c_i| - |\{x_i \,|\, x_i \in c_j \,,\, c_j \in \mathcal{L} \,,\, c_j \subset c_i\}|. \tag{5}$$

Since exploitation is based on $\mathcal{L}$, the ξ values change after each labeling iteration.

3.4.3 Exploration

Exploration spreads group labels throughout $\mathcal{H}$ to better explore the feature space as a way of discovering groups that represent concepts that have yet to be labeled. Exploration values iteratively change throughout the labeling process because they are computed with respect to $\mathcal{L}$. Specifically, the exploration score for cluster c_i is the shortest path in the hierarchy between it and the nearest labeled cluster,

$$\phi(c_i) = \min_{c_j \in \mathcal{L}} \text{ path-length}(c_i, c_j), \tag{6}$$

where the path length between c_i and c_j is the combined number of edges traversed by each node to reach their first common ancestor. For example, c_l and c_r in Fig. 5 have a path length of 2 where their first common ancestor is c. Exploration ordering labels clusters with the longest path length first (i.e., groups that are least similar to groups already discovered and labeled).

3.4.4 Multi-Objective Combination

The 3 heuristic criteria are designed to emphasize different objectives when collecting labeled training data for supervised classifiers. Since each objective is important to the problem, we define a multi-objective, rank-based combination criterion. In this combination, the set of unlabeled groups, $\mathcal{U}$, are ranked according to the 3 criteria individually, and the rankings are linearly combined to produce a multi-

objective ranking score for each cluster c_i:

$$\begin{aligned}\psi(c_i) = \beta_1 \operatorname{rank}(\Delta(c_i), \{\Delta(c_j) \,|\, c_j \in \mathcal{U}\}) \\ + \beta_2 \operatorname{rank}(\xi(c_i), \{\xi(c_j) \,|\, c_j \in \mathcal{U}\}) \\ + \beta_3 \operatorname{rank}(\phi(c_i), \{\phi(c_j) \,|\, c_j \in \mathcal{U}\}).\end{aligned} \tag{7}$$

Each β_i is a weight for its ordering objective such that $\beta_1 + \beta_2 + \beta_3 = 1.0$. For all experiments in this report, the objectives are weighted evenly, $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$. The rank function is passed a group's score and the set of scores for all other groups in $\mathcal{U}$, and returns the group's rank with respect to the set. The group with the highest combined ranking is selected as the next labeling query.

Any one of the heuristics could be used independently to select the next group to label; however, *exploitation* and *exploration* selections are not very interesting on their own. Exploitation essentially performs a breadth first search of $\mathcal{H}$, and exploration will mostly choose leaf nodes in $\mathcal{H}$ as these produce the longest path lengths. To allow the technique to generalize to any of these selection heuristics, the selection criteria is not performed on the entire unlabeled set, but a set of the most interesting groups from $\mathcal{H}$, denoted as $\mathcal{S}$. This set $\mathcal{S}$ is constructed by comparing a group's interestingness score to the distribution statistics of all interestingness scores from unlabeled groups. Formally,

$$\bar{\Delta} = \frac{1}{|\mathcal{U}|} \sum_{\forall c_i \in \mathcal{U}} \Delta(c_i) \tag{8}$$

$$\sigma_\Delta = \sqrt{\frac{1}{|\mathcal{U}|} \sum_{\forall c_i \in \mathcal{U}} (\Delta(c_i) - \bar{\Delta})^2}. \tag{9}$$

We refer to the groups with structural change values at least one standard deviation beyond the mean as outliers, and thereby, the most interesting set of groups:

$$\mathcal{S} = \{c_i \,|\, \Delta(c_i) > \bar{\Delta} + \sigma_\Delta\}. \tag{10}$$

Algorithm 1 summarizes the generalized iterative HCGL framework, where the *selection-criterion* requirement defined on line 1 could represent any of the following: HCGL-Interestingness (Eq. 4), HCGL-Exploitation (Eq. 5), HCGL-Exploration (Eq. 6), or HCGL-Combined Ranks (Eq. 7).

Algorithm 1 Hierarchical cluster guided labeling

Require: $\mathcal{H}$, selection-criterion = $\{\Delta, \xi, \phi, \psi\}$
1: $\mathcal{U} = \{c_i \mid c_i \in \mathcal{H}\}$
2: **while** $\mathcal{U} \neq \emptyset$ && user==True **do**
3: threshold = $\bar{\Delta} + \sigma_{\Delta}$
4: $\mathcal{S} = \{c_i \mid \Delta(c_i) >$ threshold, $c_i \in U\}$
5: Update selection-criterion scores $\forall c_i \in \mathcal{U}$
6: $\mathcal{S} = sort(\mathcal{S}$, selection-criterion)
7: label query $\rightarrow \mathcal{S}[0]$
8: Update $\mathcal{L}$ and $\mathcal{U}$

3.5 HCGL for Multi-Concept Images

In the context of semantic scene labeling, every pixel in a multi-concept image needs to be assigned a label. Since HCGL clusters data under the assumption that each training sample represents a single concept, images must be segmented into multiple regions. We oversegment images into approximately 150 regions using simple linear iterative clustering (SLIC).[52] Oversegmentation is performed to ensure that true region boundaries are observed in the training samples. Each segment becomes a training sample that is described by a feature vector composed of LAB color histograms, local binary patterns,[53] a 200-dimensional codebook of scale-invariant feature transform (SIFT) descriptors,[54] and normalized region coordinates. The resulting feature vectors are then hierarchically clustered using agglomerative clustering. For all experiments, we use Ward's linkage and Euclidean distance to create the hierarchy. An illustration of the HCGL framework running on image regions is shown in Fig. 6. Node colors map to a class in the label set $\mathcal{Y}$ and black wedges represent the percentage of noise in each cluster (images not representing the dominating class).

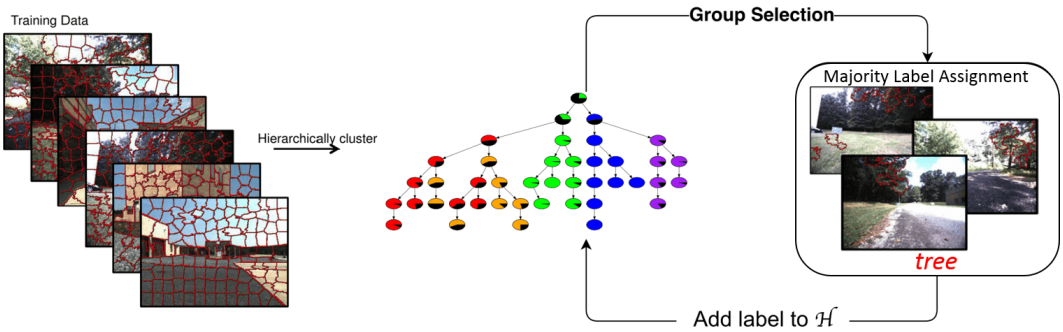


Fig. 6 Visualization of the HCGL process on multi-concept environment data

4. Speed and Classification Experiments

To demonstrate the speed and real-world feasibility of HCGL, we present results of experiments labeling real-world data in outdoor environments. Specifically, we present results showing pixel labeling rate and classification accuracy as a function of interaction time, where interaction time is the total time a human spends waiting for and answering labeling queries from a system. Thus, interaction time includes any latency introduced by techniques that recluster or retrain classifiers. This evaluation is motivated by autonomous mobile robotics applications that frequently change domains or environments, and need to quickly train classifiers to learn new terrains and objects with relatively low operator overhead.

We use 3 real-world environments to demonstrate the speed and performance of HCGL when collecting labels for multi-concept visual perception. The environments are outdoor urban training facilities that include multiple types of terrain, buildings, cars, and other objects. Training data for environment A were collected with a high dynamic range camera. Images were taken at 5 different time blocks over 2 days from 53 locations in the environment.[55] Training data for environment B and C were captured via teleoperation using the robot described in Section 5. The training set from Environment B is the combination of data collected on 3 consecutive days and is therefore much larger than the other sets. Performance on this data set shows how HCGL scales with increasing training set sizes. An overview of the data sets is provided in Table 1 and example images are provided in Fig. 7.

Table 1 Details of the real-world environment data sets

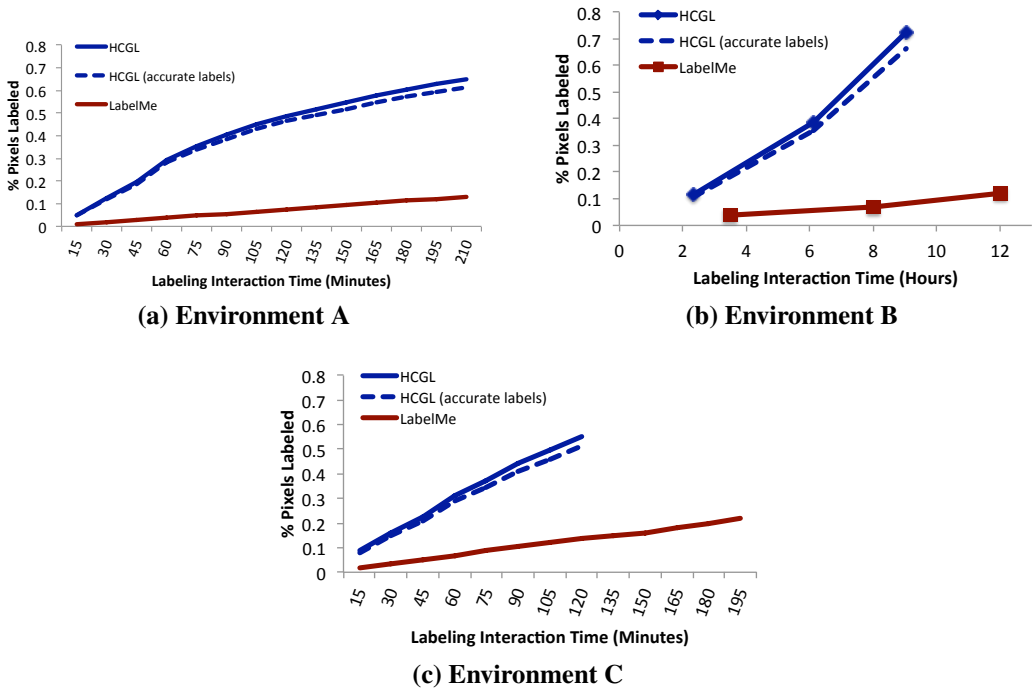
Environment	No. Training images	Label set ($\mathcal{Y}$)
A	274	*asphalt, building, concrete, grass, gravel, object, sky, tree*
B	1,982	*building, car, grass, object, road, sidewalk, sky, tree*
C	268	*building, car, curb, grass, object, road, sidewalk, sky, tree*

Fig. 7 Example images from the 3 environment training sets

For these experiments, we compare HCGL to the supervised labeling baseline, LabelMe, where training images are labeled in random order. We do not make direct comparisons to other existing efficient labeling frameworks because most frameworks do not provide interfaces for real-time labeling. We hypothesize that the latency introduced by data reclustering may inhibit the real-world practicality of such interfaces. Pixel-wise labeling and classification accuracy are evaluated as a function of labeling interaction time (i.e., adaptation latency) to show the speed at which techniques can collect multi-concept scene labels for visual classifiers.

4.1 Labeling Speed and Label Accuracy

The first experiment looks at how fast HCGL and LabelMe assign labels to pixels in the training images. Figure 8 shows the percentage of labeled pixels as a function of time. There is a large performance gap seen across all data sets. Given a fixed training time, HCGL collects around 6 to 7 times the amount of label information as LabelMe. Labeling interaction time for environment B is on the order of hours because each of the 3 days of training data were labeled separately and then combined.



(a) Environment A **(b) Environment B**

(c) Environment C

Fig. 8 Comparison of labeling rate using HCGL and LabelMe on the 3 data sets. Dashed blue lines depict the percentage of labeled pixels that received correct labels from HCGL

Assigning labels quickly is important, but recall that to achieve this speed, HCGL incurs some label noise as a result of majority labeling. The dashed blue lines in the plots show the percentage of pixels that received accurate labels from HCGL (determined using the labels collected by LabelMe). This line represents approximately 5%-10% pixel label noise: a small fraction for a large gain in efficiency.

4.2 Pixel-Wise Classification

Next, to test environment perception using labels from HCGL and LabelMe, we train a Hierarchical Inference Machine (HIM),[16] an approach for scene parsing and region classification. HIMs incorporate both feature descriptors and contextual cues computed at multiple scales within the scene. Images are decomposed into a hierarchy of nested superpixel regions,[56] where regions at the bottom provide localized discriminative information and those at the top provide global context. The predictor is a decision forest regressor with 10 trees. Features extracted from superpixels include SIFT descriptors,[54] LAB colorspace statistics, texture information, and statistics on the size and shape of superpixel regions.

4.2.1 Environment A

Pixel-wise classification accuracy is compared on a testing set from Environment A, which consists of 265 images. This is the only environment data set with a large, hand-labeled testing set.[55] Classification evaluation is performed incrementally after every 15 min in which a user assigns labels to the training set. Figure 9 shows the overall pixel accuracy for environment A. Even though HCGL introduces small amounts of label noise, the larger volume of labeled training data allows HCGL to train higher performing HIMs than LabelMe through 210 min of labeling interaction. HCGL labeling is terminated at this point to depict scenarios with limited time to devote to label collection.

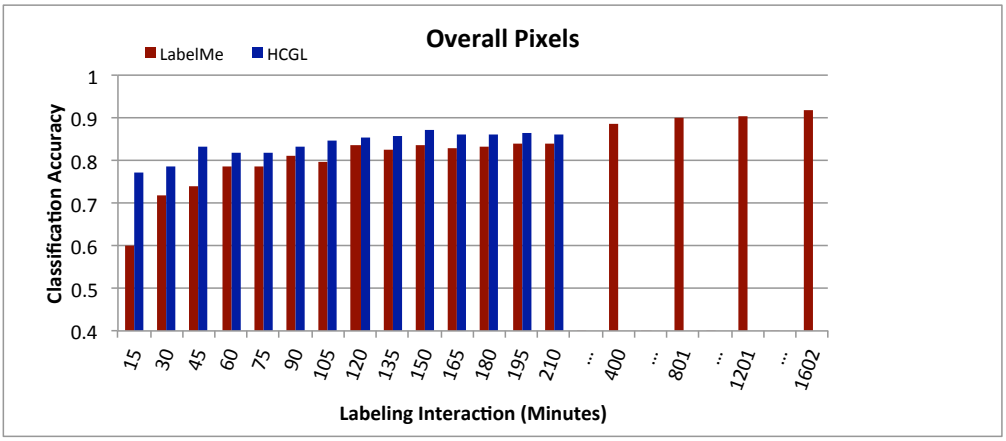


Fig. 9 Comparison of the overall pixel classification accuracy for environment A

Overall pixel classification accuracy can be skewed by classes with more pixels (pixel distributions can be seen in Fig. 10). Thus, we also evaluate per-class classification accuracy and find that HCGL performs similarly or better than LabelMe for all classes but one, as shown in Fig. 11. The *object* class is the least represented in the data and is composed of many diverse things (e.g., light poles, traffic cones, and cargo boxes). The low intraclass similarity makes it difficult for samples to group together in HCGL. Instead, samples of *object* are dispersed almost randomly across the hierarchical clustering and are often mislabeled as part of clusters dominated by other classes. As a result, HCGL achieves lower accuracy than LabelMe on this class. However, this is a poorly defined "other" class, and is difficult for LabelMe as well. With a fully labeled training set (1,602 min), LabelMe achieves only approximately 18% classification accuracy for the *object* class.

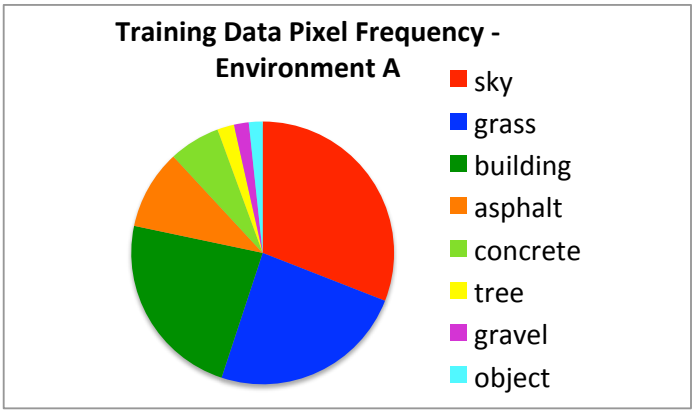


Fig. 10 Breakdown of class distributions across all pixels in the training set of environment A

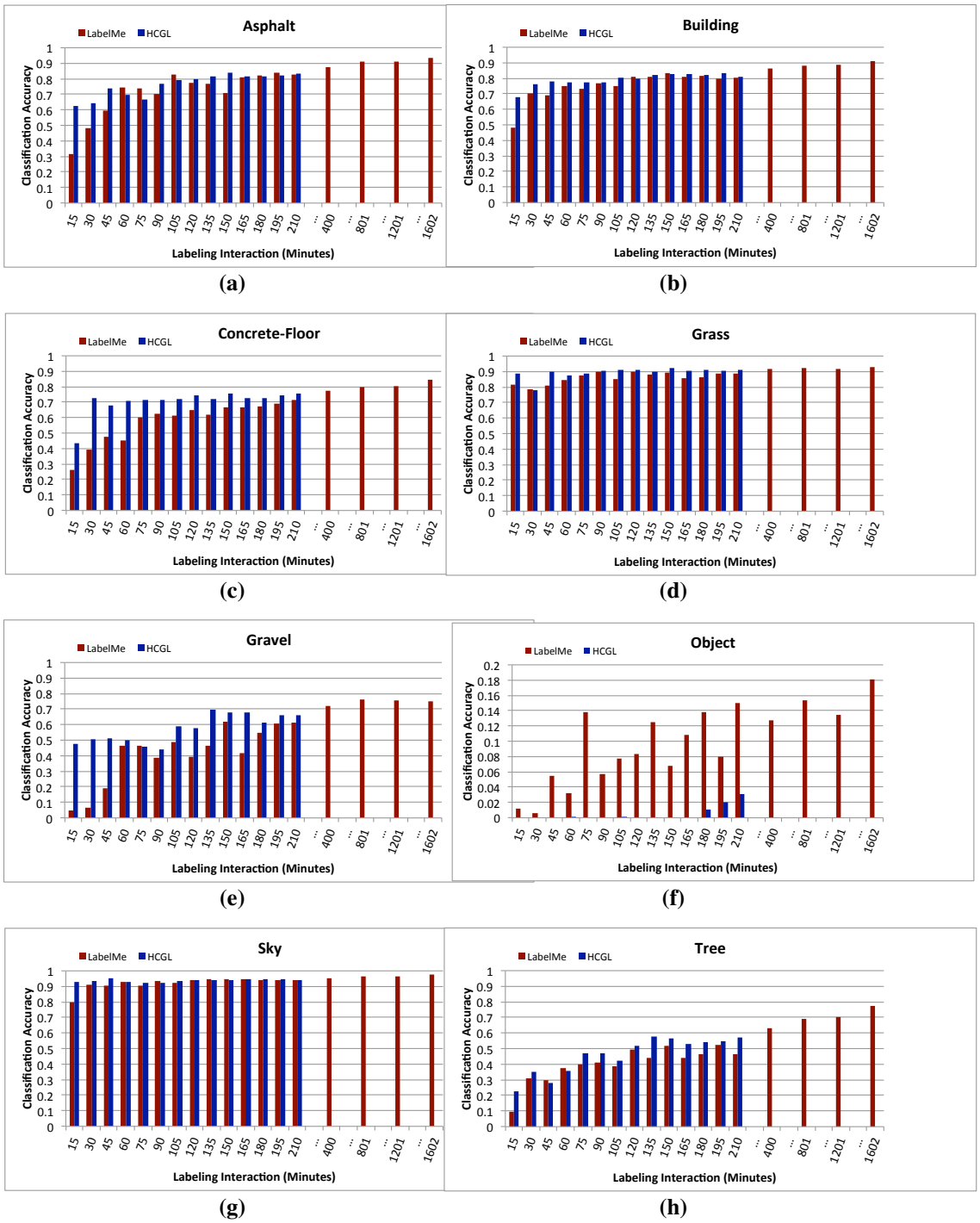
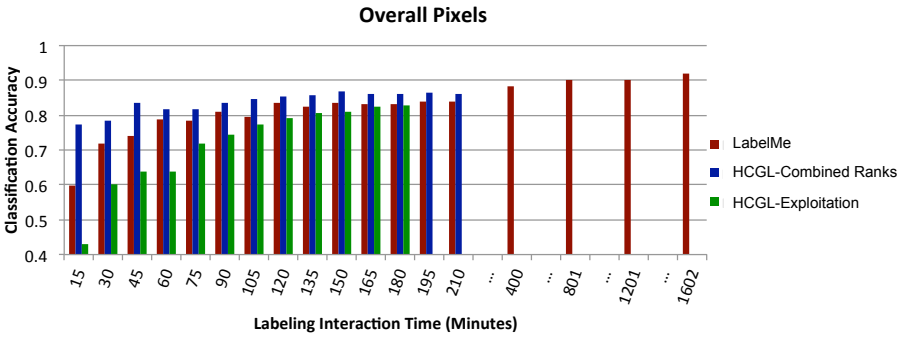


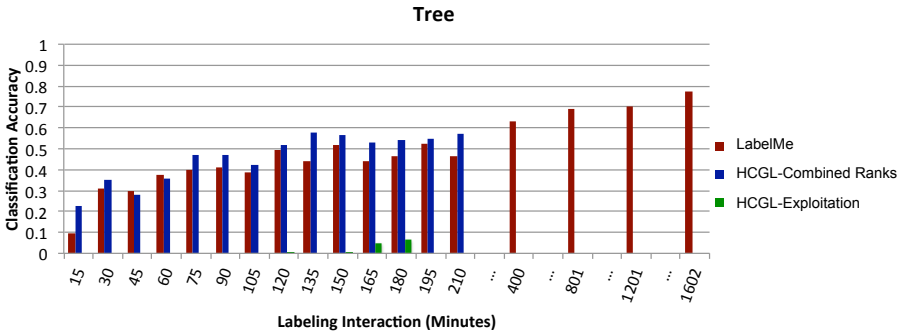
Fig. 11 Comparison of class-specific classification accuracy for environment A

In applications with limited time for label collection, it can be tempting to run HCGL with the exploitation heuristic only (Eq. 5) to collect as many labels as possible in the allotted time. We compare HCGL-exploitation and HCGL-combined ranks to once again show the importance of balancing all ordering criteria during the labeling process even under applications with limited labeling time.

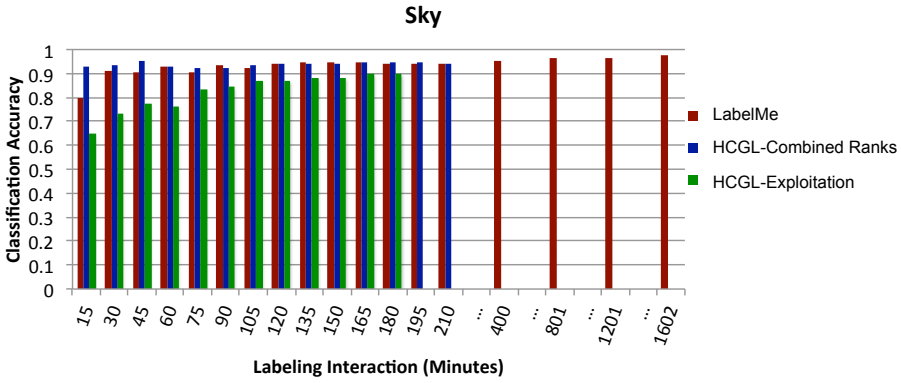
As designed, HCGL-exploitation focuses on labeling large groups of data quickly. This results in a small number of classes receiving a large number of labels early in the labeling process because of skewed class distributions. Most of the labeled training samples represent either *sky*, *grass*, or *building*. This ultimately produces worse HIM classifiers than HCGL-combined ranks and LabelMe. A subset of classification results are shown in Fig. 12, and Fig. 13 shows some examples of classified images from the test set.



(a) Overall pixels



(b) Class with small distribution of pixels.



(c) Class with large distribution of pixels.

Fig. 12 Comparison of classification accuracy for environment A using the HCGL-exploitation selection heuristic

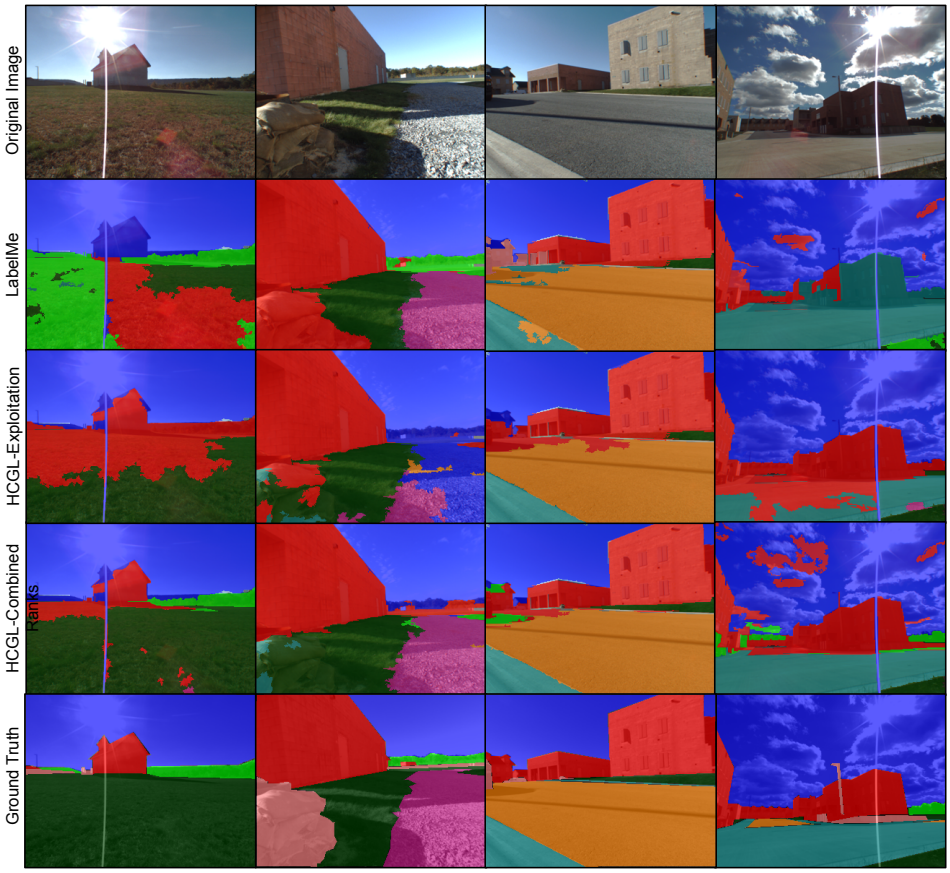


Fig. 13 Examples of classified test images from environment A that illustrate the weaknesses of HCGL-exploitation and LabelMe compared to HCGL-combined ranks

4.3 Environment B

Labeled test sets from environments B and C are not available, but we provide a qualitative pixel-wise classification comparison of HCGL and LabelMe. Figure 14 shows 6 example test images from environment B. We use LabelMe to create ground truth for these images, seen in the bottom row. Classifiers are trained using labeled data at the third markers from Fig. 8b. The selected examples show 2 instances where the classifiers perform similarly, an example where HCGL performs slightly worse than LabelMe (column 3), and the last 3 columns are examples of HCGL's superior performance and illustrate the common mistakes made by the classifier trained using LabelMe. Specifically, the LabelMe classifier often misidentifies terrain further from the camera. This allows robots to make immediate decisions, but negatively impacts long-term path planning. Qualitatively it can also be seen that HCGL commonly misclassifies *trees* and certain *objects* as *sky*, which are less costly for our navigation task. These mistakes occur because the *tree* and *object* classes are less represented than *sky* in the training set, so fewer examples are collected by HCGL. However, the overall HCGL performance on these classes is still qualitatively high. Overall, HCGL collects significantly more label information even with 25% less human interaction time and trains higher-performing classifiers.

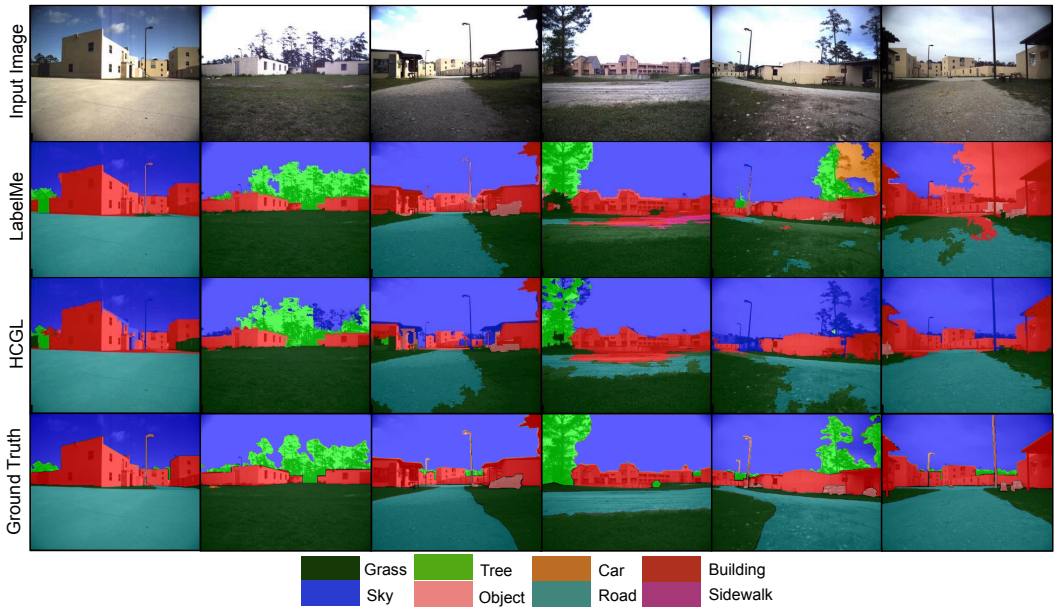


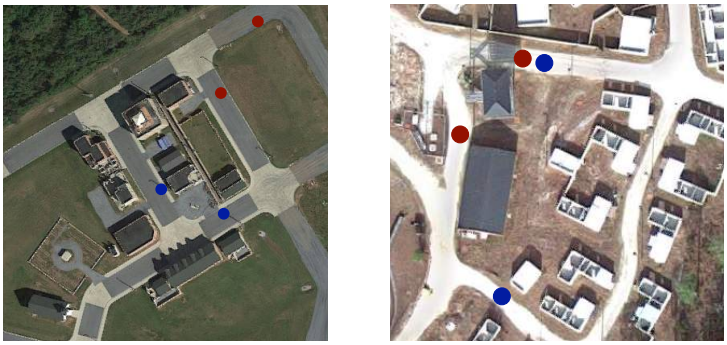
Fig. 14 Qualitative comparison between HCGL and LabelMe with a test set from environment B

5. Real-Time Navigation Experiments

Pixel-wise accuracy quantitatively compares techniques on static data, but task-based evaluation judges perception relative to the end goal of successful navigation in outdoor environments. We compare several visual classifiers trained using labels collected by HCGL and LabelMe based on their ability to provide perception information to a real-time mapping and navigation framework.[57]

5.1 Task Description

Our live navigation task requires a robot to use visual perception to plan paths between waypoints using specified terrain. These terrains are defined based on the composition of the road at testing locations. We use road traversal because roads are designed to provide navigation guidance to vehicles. For example, roads direct vehicles around buildings and hazards like bodies of water. Our experiments emulate these scenarios by defining waypoints (seen in Fig. 15) such that the most direct path to goals is not along a road.



(a) Environment A **(b) Environment B**

Fig. 15 Navigation waypoint maps for environments

Classifiers are compared based on successes and failures during multiple trials of the navigation task, where outcomes are defined as follows:

- **Success**: the robot autonomously traverses between waypoints using only road terrain without hitting objects.
- **Success with minor errors**: the robot traverses between waypoints but either 1) traverses on non-road terrain for a short duration or 2) requires operator intervention at least once but no more than twice for small adjustments in location or direction due to potential object collision or planner failure.

- **Failure**: the robot cannot plan and execute a road traversal even with minimal operator intervention, visual perception has significant false-positive errors indicating no road path, or constant planner updates result in no progress toward the goal.

5.2 Hardware

The robot used in this work, the Clearpath Husky seen in Fig. 16, is a $39 \times 26 \times 14$ inch wheeled platform that is limited to a maximum velocity of 1 m/s. The Husky employs a MicroStrain 3DM-GX3-25 IMU, a Garmin 18 GPS, and 2 Quad-Core Intel i7 Mini-ITX processing payloads, each with a 256-GB SSD running Ubuntu 14.04, ROS Indigo, and experimental software. The Husky has a Velodyne HDL-32E LiDAR, which generates $360°$ point clouds at a range of 70 m and an accuracy of up to ± 2 cm. Finally, the Husky collects image data using a Prosilica GT2750C, a 6-megapixel charge-coupled device (CCD) color camera.

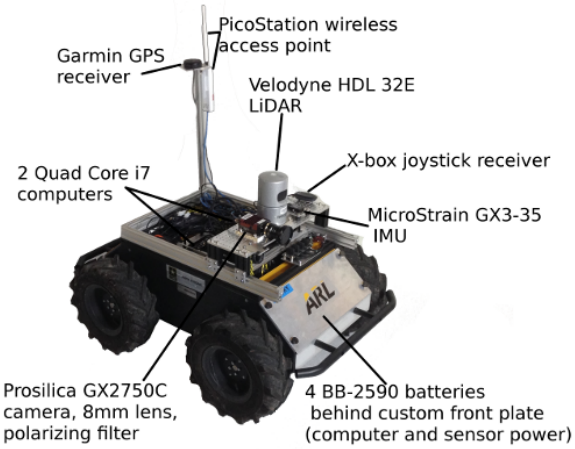


Fig. 16 Hardware configuration of the Clearpath Husky robot

5.3 Mapping and Navigation

Our robot test platform employs a mapping and navigation system to enable accurate motion between desired waypoints. The mapping system, dubbed *OmniMapper*, consumes measurements from LiDAR for relative motion estimation and loop closure through integrated color pixel (ICP),[58] GPS measurements,[59] and camera images. A keyframe is created with each measurement as the robot moves through its environment; the robot's pose at this keyframe is optimized through *GTSAM*[60] to minimize residual error from all measurements.

A 2-D local occupancy grid is created from each laser-scan keyframe through ray-tracing, where sufficient height above the ground is registered as an obstacle. When a new keyframe is added or when a significant update is made to the map causing keyframe poses to change, the 2-D occupancy grids are composited together into a negative log-odds grid and thresholded into an obstacle map as in Fig. 17b.

A keyframe is also created for each classified image, and the pose of this record is updated with the mapping process such as with loop closures or GPS measurements. Whenever a new obstacle map is created, additional cells are marked as "obstacle" if those cells, when projected into classified images, overlap with pixels classified as one of the defined non-road terrains or an object class. In Fig. 17a, only *asphalt* and *concrete* make up the road for this testing location.

(a) Environment **(b) LiDAR map** **(c) Vision map**

Fig. 17 Example obstacle maps for location 2 in the environment A. Darker regions indicate obstacles and non-road terrain.

The corners of each map grid cell (10×10 cm) are projected into all classified images that observe that cell within a range of 7 m. The classified images are rectified so the projected corners define a quad in the classified image. Each pixel in the projected quad has a label from the classifier and votes for that class to be applied to the ground cell. The ground cell is assigned the label with the highest number of votes. If this label does not represent road for navigation, the occupancy grid cell is given an obstacle value to prevent traversal through that cell. As seen in Fig. 17c, visual perception helps produce cost maps with specific terrain information (e.g., gravel regions are darker and avoided during path planning; discussed further in Section 5).

A kinematically feasible path is computed from the robot's current location to the

goal location using the Search-Based Planning Library (SBPL)[61] using a set of motion primitives generated to match the Husky's kinematics. A smoothed local plan is chosen that follows the global plan closely while avoiding local obstacles not yet present in the global map. Planner failures occur if the occupancy grid prohibits an obstacle free path to the goal. This occurs in our experiments due to false-positive non-road classifications on road terrain. See Gregory et al.[62] for more implementation details of the mapping and navigation systems used in this work.

5.4 Navigation Results: Environment A

Environment A is the primary location used for comparative evaluation since LabelMe was used to label its entire training set.[55] Four classifiers are trained and compared. We compare the labeling techniques given the same amount of labeling interaction time. **HCGL-150** and **LabelMe-150** represent classifiers trained after 150 min of labeling, which reflects scenarios where limited labeling time is available. This is just under one-tenth of the estimated total time (1,602 min) required to label the entire training set with LabelMe. To demonstrate results given no time restrictions, a classifier is trained using the entire training set, denoted as **LabelMe-1602**.

The final classifier is meant to show the benefits of using training data representing the most recent state of a robot's environment and how HCGL easily facilitates the labeling of data upon arrival to a new or changed environment. We supplement the existing training set (collected several years ago) with 231 additional images collected during our experiments (disjoint from testing locations). Labeling was performed for 30 min with HCGL, and approximately 27% of the pixels in the new images were assigned labels. Without ground truth for this set, the amount of collected label noise is unknown. This set of labeled data is combined with the labeled data from HCGL-150 to train the final classifier, denoted as **HCGL-150+30**.

Navigation experiments are performed at 2 locations in the environment. Location 1 is illustrated with red waypoints in Fig. 15a, and roads are composed of *gravel*, *concrete*, and *asphalt*. Thus, path planning must avoid grass terrain (the shortest path between waypoints) and several objects near the edge of the grass and road. Each trial represents a traversal from one waypoint to the other and are performed in both directions. Trials were run across multiple days and different times of day to capture performance under varying environment conditions. Table 2 compares the

performance of each classifier at this first location.

Table 2 Summary of navigation results for location 1 (red waypoints) in environment A

	% Successes		
Label Model	No Errors	Minor Errors	% Failures
HCGL-150	0.500	0.000	0.500
LabelMe-150	0.333	0.167	0.500
LabelMe-1602	0.250	0.250	0.500
HCGL-150+30	**0.875**	**0.125**	0.000

HCGL-150 and LabelMe-150 perform similarly and inconsistently with a 50% failure rate. LabelMe-1602 exhibits the same failure rate, but also displays more minor errors during its successful trials. LabelMe-1602 uses the most labeled data to learn class boundaries with respect to the training set, but performs worse because the learned class boundaries changed. The classifiers trained after 150 min likely learned less definitive class boundaries, making the environment changes less detrimental. Some observed changes from the training data include grass length, cloud coverage, and illumination. HCGL-150+30, on the other hand, performs the navigation task very reliably, because it represents a classifier that has adapted to the changed environment with new and additional training data. Minor errors involved the robot trying to plan a shortest path through the grass, entering the grass for a brief moment before backing out, and successfully planning a road traversal route. These results demonstrate the positive impact of rapid label collection, even if a small fraction is noisy, when new training data are needed to adapt and improve visual perception.

Qualitative evaluation of visual perception shows the labeling models produce classifiers that make different mistakes. Figure 18 includes examples explicitly chosen to depict some of the worst classified images by one or more models. HCGL-150 had many false-positive *concrete* classifications, which can be seen best in columns 1, 3 and 4. Columns 3 and 5 highlight that LabelMe-150 produced more false-positives of *object* and *building* classes on what was actually road terrain. LabelMe-1602 has cleaner results than the previous models, but also often misclassified *gravel* as *object* (seen in column 3), and tended to misclassify *trees* as *buildings* (seen in columns 1 and 2). Although still not perfect classification, HCGL-150+30 has the most accurate results compared to the ground truth, which yielded its superior navigation success and highlights the importance of being able to quickly collect large amounts of new labeled training data given environment changes.

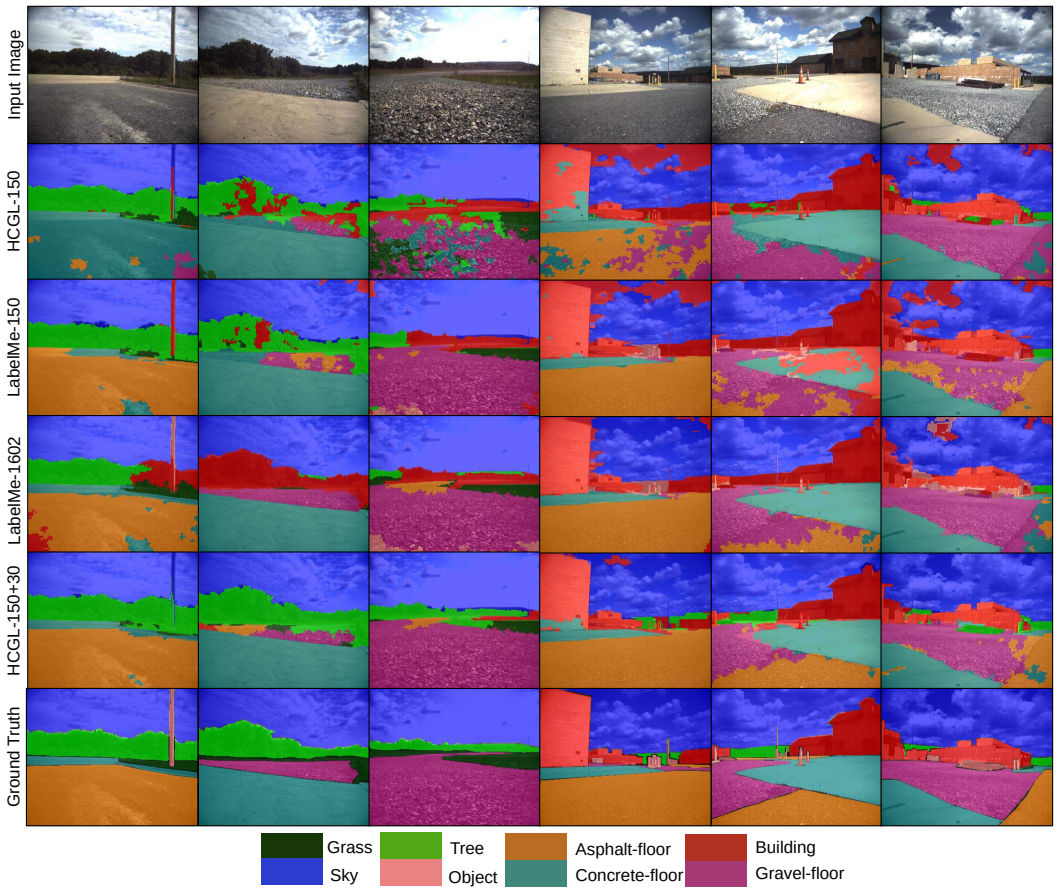


Fig. 18 Visual perception examples for each labeling model. Images in the first 3 columns are from the first location (red waypoints), and the last 3 columns are images from the second location (blue waypoints)

The second location is depicted in Fig. 15a with blue waypoints. At this location, roads are composed of *concrete* and *asphalt*, whereas *gravel* terrain (shortest path between waypoints) is not road. Along the shortest road path are 2 objects (traffic cones) that the robot must also avoid. Terrain classification for classes with high interclass similarity is important for successful traversal during this test.

Comparisons are made between LabelMe-1602 and HCGL-150+30; the most successful models at the first location in terms of successes and qualitative evaluation. Results are summarized in Table 3 and indicate that this navigation task is more challenging. However, HCGL-150+30 is still able to successfully navigate the majority of the time with only minor errors. Most failures and errors at this location were caused by classification confusion of *asphalt* and *gravel*. This can be seen in the last 3 columns of Fig. 18.

Table 3 Summary of navigation results for location 2 (blue waypoints) in environment A

	% Successes		
Label Model	No Errors	Minor Errors	% Failures
LabelMe-1602	0.000	0.000	1.000
HCGL-150+30	**0.375**	**0.250**	0.375

5.5 Navigation Results: Environment B

We use environment B, seen in Fig. 15b, to further test HCGL label collection in new domains. In this environment, roads are composed of a single terrain type labeled as *road*, and all other terrains and objects should be avoided during path planning. Training data for this environment were not available prior to the experiment, so data were collected upon arrival. We chose to focus our navigation trial experiments on labels collected using HCGL to show the consistency of the system across multiple environments.

An in-depth discussion and analysis of experiments in this environment is omitted, but examples of the robot perception in this environment were seen in Fig. 14. Over 15 navigation trials were performed between both waypoint sets without any failure cases. Only minor path planning errors in a few trials caused the robot to traverse on the edge of the *grass* where it meets the *road*. These successes are used to confirm that small amounts of label noise collected by HCGL, in exchange for fast label collection, does not negatively impact path planning.

6. Conclusion and Future Work

Real-time visual perception for mobile robots is only as useful as its ability to quickly adapt to changing environments. We discussed an efficient label collection technique, HCGL, for multi-concept environment data. It was shown that while HCGL trades some label accuracy for reduced adaptation latency, this label noise does not significantly impact visual perception for navigation. Using this technique, high-quality visual perception can be obtained in new environments with only a few hours of labeling effort from a human annotator.

The multi-concept semantics provided by HCGL allow this work to generalize to more complex variations of path planning tasks. This includes assigning variable costs to terrains based on robot capabilities and path planning with verbal navigation cues given during human-robot interaction. Future work also includes augment-

ing HCGL to be even more effective through online label collection and adaptation.

7. References

1. Häselich M, Arends M, Wojke N, Neuhaus F, Paulus D. Probabilistic terrain classification in unstructured environments. Robotics and Autonomous Systems. 2013;61(10):1051–1059.

2. Kümmerle R, Ruhnke M, Steder B, Stachniss C, Burgard W. Autonomous robot navigation in highly populated pedestrian zones. J Field Robotics. 2015;32(4):565–589.

3. Milella A, Reina G, Underwood J. A self-learning framework for statistical ground classification using RADAR and monocular vision. J Field Robotics. 2015;32(1):20–41.

4. Manjanna S, Dudek G, Giguere P. Using gait change for terrain sensing by robots. In: Int Conference on Computer and Robot Vision; 2013; Saskatchenwan, Canada. p. 16–22.

5. Khan YN, Masselli A, Zell A. Visual terrain classification by flying robots. In: Int Conference on Robotics and Automation; 2012; St. Paul, MN. p. 498–503.

6. Filitchkin P, Byl K. Feature-based terrain classification for littledog. In: Int Conference on Intelligent Robots and Systems; 2012; Vilamoura, Portugal. p. 1387–1392.

7. Zenker S, Aksoy EE, Goldschmidt D, Worgotter F, Manoonpong P. Visual terrain classification for selecting energy efficient gaits of a hexapod robot. In: Int Conf on Advanced Intelligent Mechatronics; 2013; Novotel Wollongong, Australia. p. 577–584.

8. Hoffmann M, Štěpánová K, Reinstein M. The effect of motor action and different sensory modalities on terrain classification in a quadruped robot running with multiple gaits. Robotics and Autonomous Systems. 2014;62(12):1790–1798.

9. Roncancio H, Becker M, Broggi A, Cattani S. Traversability analysis using terrain mapping and online-trained terrain type classifier. In: Intelligent Vehicles Symposium; 2014 ; Dearborn, MI. p. 1239–1244.

10. Ross P, English A, Ball D, Upcroft B, Corke P. Online novelty-based visual obstacle detection for field robotics. In: Int Conference on Robotics and Automation; 2015; Seattle, WA. p. 3935–3940.

11. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems; 2012; Lake Tahoe, NV. p. 1097–1105.

12. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Computer Vision and Pattern Recognition; 2015 June; Boston, MA.

13. Deng J, Dong W, Socher LJ Richard Li, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition; 2009; Miami Beach, FL.

14. Fei-Fei L. Crowdsourcing, benchmarking and other cool things. 2010 [accessed 2016 Sep 21]. http://www.image-net.org/papers/ImageNet_2010.pdf.

15. Farabet C, Couprie C, Najman L, LeCun Y. Learning hierarchical features for scene labeling. Trans Pattern Analysis and Machine Intelligence. 2013;35(8):1915–1929.

16. Munoz D. Inference machines: parsing scenes via iterated predictions [thesis]. The Robotics Institute, Carnegie Mellon University; 2013.

17. He H, Upcroft B. Nonparametric semantic segmentation for 3D street scenes. In: Int Conference on Intelligent Robots and Systems; 2013; Tokoyo, Japan. p. 3697–3703.

18. Lee YJ, Grauman K. Learning the easy things first: Self-paced visual category discovery. In: Computer Vision and Pattern Recognition; 2011; Colorado Springs, CO. p. 1721–1728.

19. Chen J, Cui Y, Ye G, Liu D, Chang SF. Event-driven semantic concept discovery by exploiting weakly tagged internet images. In: Int Conference on Multimedia Retrieval; 2014; Glasgow, Scotland. p. 1.

20. Sun C, Gan C, Nevatia R. Automatic concept discovery from parallel text and visual corpora. In: Int Conference on Computer Vision; 2015; Santiago, Chile. p. 2596–2604.

21. Kapoor A, Grauman K, Urtasun R, Darrell T. Active learning with gaussian processes for object categorization. In: Int Conference on Computer Vision; 2007; Rio de Janeiro, Brazil. p. 1–8.

22. Holub A, Perona P, Burl MC. Entropy-based active learning for object recognition. In: Computer Vision and Pattern Recognition Workshops; 2008; An-chorage, AK. p. 1–8.

23. Jain P, Kapoor A. Active learning for large multi-class problems. In: Computer Vision and Pattern Recognition; 2009; Miami Beach, FL. p. 762–769.

24. Joshi AJ, Porikli F, Papanikolopoulos N. Multi-class active learning for image classification. In: Computer Vision and Pattern Recognition; 2009; Miami Beach, FL. p. 2372–2379.

25. Li X, Guo Y. Adaptive active learning for image classification. In: Computer Vision and Pattern Recognition; 2013; Portland, OR.

26. Tuytelaars T, Lampert CH, Blaschko MB, Buntine W. Unsupervised object discovery: A comparison. Int J Computer Vision. 2010;88(2):284–302.

27. Dai D, Prasad M, Leistner C, Van Gool L. Ensemble partitioning for unsupervised image categorization. In: European Conference on Computer Vision; 2012 ; Firenze, Italy. Springer; 2012. p. 483–496.

28. Lee YJ, Grauman K. Object-graphs for context-aware visual category discovery. Trans Pattern Analysis and Machine Intelligence. 2012;34(2):346–358.

29. Galleguillos C, McFee B, Lanckriet G. Iterative category discovery via multiple kernel metric learning. Int J Computer Vision. 2014;108(1-2):115-132.

30. Biswas A, Jacobs D. Active image clustering: Seeking constraints from humans to complement algorithms. In: Computer Vision and Pattern Recognition; 2012; Providence, RI. p. 2152–2159.

31. Xiong C, Johnson DM, Corso JJ. Spectral active clustering via purification of the k-nearest neighbor graph. In: European Conference on Data Mining; 2012; Libson, Portugal.

32. Liu D, Chen T. Unsupervised image categorization and object localization using topic models and correspondences between images. In: Int Conference on Computer Vision; 2007; Rio de Janeiro, Brazil. p. 1–7.

33. Sivic J, Russell B, Efros A, Zisserman A, Freeman W. Discovering objects and their location in images. In: Int Conference on Computer Vision; 2005 Oct; Beijing, China. p. 370–377.

34. Kim D, Sun J, Oh SM, Rehg JM, Bobick AF. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In: Int Conference on Robotics and Automation; 2006; Orlando, FL. p. 518–525.

35. Dahlkamp H, Kaehler A, Stavens D, Thrun S, Bradski GR. Self-supervised monocular road detection in desert terrain.. In: Robotics: Science and Systems; 2006; Philadelphia, PA.

36. Lookingbill A, Rogers J, Lieb D, Curry J, Thrun S. Reverse optical flow for self-supervised adaptive autonomous robot navigation. Int J Computer Vision. 2007;74(3).

37. Zhou S, Xi J, McDaniel MW, Nishihata T, Salesses P, Iagnemma K. Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain. J Field Robotics. 2012;29(2):277–297.

38. Summers-Stay D, Cassidy T, Voss CR. Joint navigation in commander/robot teams: Dialog & task performance when vision is bandwidth-limited. V&L Net. 2014;:9.

39. Sorokin A, Forsyth D. Utility data annotation with amazon mechanical turk. In: Computer Vision and Pattern Recognition Workshops; 2008; Anchorage, AK.

40. Chilton LB, Little G, Edge D, Weld DS, Landay JA. Cascade: crowdsourcing taxonomy creation. In: Conference on Human Factors in Computing Systems; 2013; Paris, France. p. 1999–2008.

41. Vijayanarasimhan S, Jain P, Grauman K. Far-sighted active learning on a budget for image and video recognition. In: Computer Vision and Pattern Recognition; 2010; San Francisco, CA. p. 3035–3042.

42. Vijayanarasimhan S, Grauman K. Large-scale live active learning: training object detectors with crawled data and crowds. Int J Computer Vision. 2014;108(1-2):97-114.

43. Gilbert A, Bowden R. igroup: Weakly supervised image and video grouping. In: Int Conference on Computer Vision; 2011; Barcelona, Spain. p. 2166–2173.

44. Xie J, Kiefel M, Sun M, Geiger A. Semantic instance annotation of street scenes by 3d to 2d label transfer. In: Computer Vision and Pattern Recognition; 2016; Las Vegas, NV.

45. Nagaraja NS, Ochs P, Liu K, Brox T. Hierarchy of localized random forests for video annotation. In: Joint DAGM and OAGM Symposium; 2012; Graz, Austria. p. 21–30.

46. Badrinarayanan V, Budvytis I, Cipolla R. Mixture of trees probabilistic graphical model for video segmentation. Int J Computer Vision. 2014;110(1):14–29.

47. Konolige K, Agrawal M, Bolles RC, Cowan C, Fischler M, Gerkey B. Outdoor mapping and navigation using stereo vision. In: Experimental Robotics; 2008; Rio de Janeiro, Brazil. p. 179–190.

48. Russell BC, Torralba A, Murphy KP, Freeman WT. LabelMe: a database and web-based tool for image annotation. Int J Computer Vision. 2008;77(1-3).

49. Wigness M, Draper BA, Beveridge JR. Efficient label collection for unlabeled image datasets. In: Computer Vision and Pattern Recognition; 2015; Boston, MA. p. 4594–4602.

50. Fei-Fei L, Perona P. A bayesian hierarchical model for learning natural scene categories. In: Computer Vision and Pattern Recognition; Vol. 2; 2005; Los Alamitos, CA. p. 524–531.

51. Griffin G, Holub A, Perona P. Caltech-256 object category dataset. Pasadena (CA): Caltech; 2007.

52. Achanta R, Shaji A, Smith K, Lucchi A, Fua P, Susstrunk S. SLIC superpixels compared to state-of-the-art superpixel methods. Trans Pattern Analysis and Machine Intelligence. 2012;34(11):2274–2282.

53. Ojala T, Pietikäinen M, Harwood D. A comparative study of texture measures with classification based on featured distributions. Pattern recognition. 1996;29(1):51–59.

54. Lowe DG. Distinctive image features from scale-invariant keypoints. Int J computer vision. 2004;60(2):91–110.

55. Lennon C, Bodt B, Childers M, Camden R, Suppé A, Navarro-Serment L, Florea N. Performance evaluation of a semantic perception classifier. Adelphi (MD) Army Research Laboratory (US); 2013. Report No.: ARL-TR-6653.

56. Felzenszwalb PF, Huttenlocher DP. Efficient graph-based image segmentation. Int J Computer Vision. 2004;59(2).

57. Wigness M, Rogers III JG, Navarro-Serment LE, Suppe A, Draper BA. Reducing adaptation latency for multi-concept visual perception in outdoor environments. In: Int Conference on Intelligent Robots and Systems; 2016; Daejeon, Korea.

58. Segal A, Haehnel D, Thrun S. Generalized-ICP. In: Robotics: Science and Sys-tems; Vol. 2; 2009; Seattle, WA.

59. Rogers JG et al.. Mapping with a ground robot in GPS denied and degraded environments. In: American Control Conference; 2014; Portland, OR. p. 1880–1885.

60. Dellaert F, Kaess M. Square root SAM: Simultaneous localization and mapping via square root information smoothing. Int J Robotics Research. 2006;25(12).

61. Cohen BJ, Chitta S, Likhachev M. Search-based planning for manipulation with motion primitives. In: Int Conference on Robotics and Automation; 2010; Anchorage, AK. p. 2902–2908.

62. Gregory J, Fink J, Stump E, Twigg J, Rogers J, Baran D, Fung N, Young S. Application of multi-robot systems to disaster-relief scenarios with limited communication. In: Field and Service Robotics; 2015; Toronto, Canada.

INTENTIONALLY LEFT BLANK.

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO L IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
3 (PDF)	US ARMY RESEARCH LAB RDRL CII A S YOUNG M WIGNESS J ROGERS
1 (PDF)	CARNEGIE MELLON UNIVERSITY L NAVARRO-SERMENT

INTENTIONALLY LEFT BLANK.